A SEARCHABLE LOGICAL DOCUMENT ARCHITECTURE

Background Of The Invention

Field of the Invention

This invention generally relates to searching computer databases, and more specifically, the invention relates to searching a collaborative document database.

Prior Art

In a collaborative document database system, there may be the concept of a logical document, where a parent document may have attached multiple children documents. Such systems do not usually store the data in these documents in a relational database. Instead, the documents are stored in a proprietary format and related documents may be only weakly linked or associated. Even though the data are not stored with the relational model, querying for individual documents is normally not a problem, as the document database system may be equipped with a full text search engine.

A problem may occur, however, when querying against a logical document as a whole. For example, a query may need to find all the criminal investigations where "John Doe" and "Tom Smith" were involved. With reference to Figure 1, if "John Doe" was located in the data for Suspect 1 and "Tom Smith" was in the data contained by the Suspect 2 child document, the query would fail. It would not find the logical document that contained both of these suspects.

Summary Of The Invention

An object of this invention is to provide a method and system that supports the searching for data within a complete logical document.

Another object of the present invention is to allow the restriction of returned data query results based on including singleton fields from a parent document in an index document.

These and other objectives are attained with a method and system for searching a collaborative document database. The method comprises the steps of providing parent documents and child documents in said database; and for each parent document, building an index document listing a portion of said child documents. The method comprises the further steps of providing a search term, and searching all said index documents for said search term.

Preferably, each child document is related to at least one of the parent documents; and the building step includes the step of identifying in the index document for each parent document, at least some of the child documents related to the parent document. With this embodiment, the searching step may include the step of, for each parent document, searching for the search term in the child documents identified in the index document for the parent document.

Further benefits and advantages of the invention will become apparent from a consideration of the following detailed description, given with reference to the accompanying drawings, which specify and show preferred embodiments of the invention.

Brief Description Of The Drawings

Figure 1 shows a collaborative document database.

Figure 2 illustrates a searchable logical document architecture.

Figure 3 shows an index document architecture embodying the present invention.

Detailed Description Of The Preferred Embodiments

Figure 2 shows two databases 12 and 14. Database 12 contains the actual logical documents while database 14 contains index documents. The index document database is full text indexed periodically. Both databases may be on a suitable server (not shown). Further, a typical document search graphical user interface (GUI) 16 is supplied. With the particular embodiment illustrated in Figure 2, this GUI is hosted within the logical document database space, but this is not necessary to the practice of this invention. The search user interface 16 can be provided by any suitable user interface, and many suitable user interfaces are well known in the art.

Figure 3 depicts how an index document 20 may be built from a logical document 22. The preferred method used to actually create the index document is as follows:

1. Find the new or modified parent 24 or descendant documents in the document database on the server.
2. Find the index document 20 for the logical document-parent in the index database.
3. Clear the data in the index document.
4. Take selected singleton field items (such as the singleton items that are considered most important) from the logical document parent and place those field items as fields on the index document.
5. Proceed through the parent document and take all text and place that text in search text fields on the index document.
6. Proceed through the hierarchy of remaining child documents 40 in the logical document.
7. Obtain all the text data from each child and concatenate that text to the data in the search text block field on the index document.

8. Update the full text index on the index database.

Figure 2 also shows how this system may be used. As new or modified documents occur on the logical document database 12, index documents are generated and placed in the index document database 14. The index document database 14 is periodically full text indexed.

The search user interface 16 generates queries 42 that run against the index document database 14. Query results 44 are returned to the user interface 16 as a list of logical document references. When a user, as represented at 46, selects one of these references, the logical document is returned, as represented at 50, from the logical document database 12.

The specific fields that are included in the index document are useful to limit data returned from the query 22. For example, the query could be built to specify that the "Date of Crime" field can only contain dates within a certain time frame.

The present invention, as described above, provides the ability to apply a full text query expression across a collection of related documents. The following example illustrates this feature of the invention.

As an example, some kind of criminal incident occurs (say a murder). In the database 12, a document is saved for this case. Next, a witness shows up who had seen some one fleeing the scene wearing a pink dress. In the database 12, a supplementary document is saved with the witness's statement regarding the pink dress. Sometime later, some evidence is found, say 9mm shells. In the database 12, another supplementary document is saved with the evidence of the 9mm shells. Finally, there are some actual suspects. In the database 12, a supplementary document is saved with information about Joe Smith the primary suspect. Later, the case is closed because it could not be proved that Joe Smith committed the crime.

The case in the database 12 is a collection of separate documents, each reflecting some supplementary information about the case and possible corresponding to an actual paper document in the real world.

Years later, another murder occurs where a witness sees the perpetrator fleeing the scene in a pink dress and there are 9mm shells left at the scene. It would be ideal to run a full text query that says: "Give me all the cases involving a (pink near dress) and (9mm near shells)."

In a standard index as described in U.S. Patent 5,742,816, this query will not find the case described above because its database saves discrete documents and the expression (pink near dress) and (9mm near shells) does not occur within a discrete document. Rather, each term is found in a different document.

With the present invention, in contrast, this case is found. This invention achieves this by creating and updating an index document for the parent document whenever a document in its hierarchy changes. Essentially, a collection of documents is changed into a master document.

While it is apparent that the invention herein disclosed is well calculated to fulfill the objects stated above, it will be appreciated that numerous modifications and embodiments may be devised by those skilled in the art, and it is intended that the appended claims cover all such modifications and embodiments as fall within the true spirit and scope of the present invention.